Towards a Blockchain-Based Digital Copyright Contract Platform

Jingkai Chen
Tianyi Li
Daniel Whatley

May 16, 2018

Contents

Chapter 1

Introduction

Digital documents are ubiquitous among the Internet in many forms. Even after several decades, however, the copyright management of such digital documents has not followed any single copyright protocol. There are two main challenges in digital content management: users cannot authenticate digital documents since the content sources are often unknown, and providers currently have no efficient method to track article usage or identify potential copyright violations. In this report, we investigate how blockchain technology provides a potential solution to these challenges.

This blockchain-based approach has several advantages. First, the protocol is decentralized since it mitigates the throughput pressure of center servers. Second, in this system, documents are trackable and verifiable, since both users and providers can easily authenticate document identities via past additions to the blockchain. Third, the approach is considerably more flexible and extensible.

1.1 Ethereum and ERC-20

Ethereum [2] is currently the second largest blockchain system next to Bitcoin [7]. The core idea of Ethereum is based on a smart contract that is secured by a cryptographical chain structure. Unlike Bitcoin, which mainly serves as a payment medium, Ethereum is essentially a root platform supporting secondary blockchain structure development and provides abundant space for chain designs under specific scenarios. Ethereum has also established the *ERC20 token standard* [11] for smart contract applications, such as digital wallets and token exchange. The establishment of the ERC20 stardard has inspired many ICO (Initial Coin Offering) events, and pushed the blockchain innovation to a second peak at around 2014.

Many smart contract applications have been established based on the ERC20 standard in the past year. Some, including EOS, TRX, QTUM, and WTC, have become quite successful [5]. As Ethereum published detailed tutorials on how to create an ERC20-based new token [4], creating a minimalistic ERC20-based token is now a very simple and quick process, illustrating the fast development of Ethereum and ERC20 in terms of efficiency and accessibility.

An important reason of the popularity of the ERC20 standard is that it secures transactions while providing much flexibility for smart contract design. In other words, while enjoying efficient back-ends, developers can specialize the top-level designs according to specific applications. Even though many promising blockchain applications have chain structures entirely

independent of ERC20, ERC20-based token creation is still a good exercise for beginners, which motivates us to solve these copyright problems using an ERC20-based approach.

Aside from serving as a payment medium (e.g. Bitcoin), blockchains have had few practical applications. Until recently, the concepts of decentralization, verifiability, trackability, freelancing, and so on, have not yet been launched on a specific working scenario.

1.2 Copyright Chain

One recently-developed promising application of blockchain technology is the “copyright chain": utilizing the verifiable and trackable nature of the blockchain to protect the copyright of digital artifacts. This idea is promising in that it is essentially re-creating truth online that is lost among the virtual internet space. In the digital age today, people still value truth—truth is becoming harder and harder to maintain, so it is more valuable today than ever before. It is a widely-held belief that blockchain technology will help digital content management.

However, most existing projects have a simplistic structure that leave much room for improvement. Previous work has relied mostly on the timestamps of the chain and do not incorporate additional cryptographical features. To make the copyright chain more secure, a few new ideas could be implemented with cryptographical tools. For example, in existing projects, the digital content is still public; it is expected that the content could also be encrypted while in transaction in order to prevent adverse duplication.

We present the design for a new copyright smart contract with an improved structure having advanced cryptographical features. We are aiming to find solutions to fix the multiple challenges of digital content copyright. We apply a series of cryptographical tools such as MAC, block cipher, public key encryption etc. in the smart contract design. Our design follows ERC20 standard and is implemented on Ethereum.

Aiming to tackle the copyright problems, a few copyright projects have been launched [1], which we describe in Section 2.2.

Chapter 2

Motivation

2.1 Digital Rights Management

As discussed, it is not easy to achieve well-protected copyright of digital content—due to the nature of an online environment, massive diffusion of information can quickly result. Once a piece of digital work is uploaded online, it could immediately have many duplicates, some of which may even be slightly modified and hence counterfeit. On one hand, open-source promoters argue that this is exactly one advantage of the Internet: sharing and transparent. On the other hand, however, digital content providers complain that their work is not protected at all, and there is an essential distinction to be made between open source code and digital artworks. The former is extendable and thus the idea of open-source could help make consecutive progress, while the latter are rather complete and should stand as independent entities.

The value of a piece of content (either digital or hard-copy) is determined by the difficulty of access to the content. Digital content that is widely spread online, therefore, effectively have zero value. Digital rights transfers are typically one-time: the content provider uploads his work to a platform and receives a one-time profit, while renouncing the complete right of his production. From now on, the right of the content "vanishes": the platform is assumed to hold the right, but since access to the content is wide open and no one is able to track the content leakage, the copyright is lost due to anonymity.

Digital rights management focuses on creating uniqueness around digital contents: once the uniqueness of digital works is reclaimed, like the offline contents, the value of those contents will be rediscovered. As one can imagine, cryptography provides plausible solutions to create uniqueness around digital contents as well as enabling authentication of the objects and the process.

2.2 Related Work

Currently there are a few platforms that aim to get around the issue of digital rights management using cryptographic features, mostly blockchain structures. Here we list three platforms, discussing their ideas and flaws.

2.2.1 Ujo Music

Ujo Music is a design based on Ethereum. It creates an identity for each uploaded music pieces, by which it keeps track of the digital rights owner. The payments in ownership transfer and in contents play are distributed using Ethereum smart contracts. The creator of contents and subsequent owners have multiple options of controlling the content spread [10]. However, on Ujo Music the contents are not protected against duplication; there is no incentive to guarantee that the content viewers will actively stand out to protect the content copyrights.

2.2.2 Binded

Binded is another well-established digital rights management platform that is built on top of Bitcoin. It keeps track of the contents ownership using a similar chain structure as Ujo Music; moreover, it creates a permanent digital fingerprint (watermark) for each file. In this way, the duplication is assumed to be prevented; however, digital contents are still subject to anonymous attacks: it is still possible to leak the contents anonymously without being detected. Also, the removability of the fingerprint is not clear.

2.2.3 Ascribe

Ascribe creates spreading graphs on each digital original by tracking its usage. It claims to maintain the transparent visibility of the spread of contents, relying on another application *WhereOnThe.Net*. Like Ujo Music and Binded, it creates a certificate of each content file. The problem is that Ascribe assumes limited editions of files; this may place constraints on the transaction as well as promotion of digital contents.

In addition, the main incentive in Ascribe to "play by the rules" (i.e., not upload counterfeit work) is money. There is, however, nothing else preventing counterfeit data from being written to future blocks, and the monetary gain one can obtain from "selling" counterfeit work could far outweigh the monetary loss from writing a counterfeit block once.

Chapter 3

Design

3.1 Our Solution

The solution we propose to solve the problems detailed in Chapters 1 and 2 is to create uniqueness around digital content using cryptographic features. The cryptographic techinques we introduce into the blockchain structure help ensure that the value of digital content remains. However, even if digital contents have nonzero value, the critical question to be asked is: who will buy digital content off our proposed platform, and why would people want to buy? Hard-copy art works serve as collector items, and the physical nature adds to the sense of ownership. As for digital contents, however, the physical sense is lost; we therefore must figure out an incentive for people to purchase digital contents.

In most existing media platforms, all user payments are collected solely by the platform, and in same cases a small share is given to the content generator. Our solution is to incentivize ownership of digital contents by giving *all content owners*, in addition to the platform, a fixed share of the fees users pay (Figure 1). Namely, for each payment a user makes, a portion of it goes to the platform and a portion goes to the content owners—not just the original content provider. Due to these monetary incentives for content owners, in a sense a market is created for each piece of work uploaded.

3.2 Blockchain Design

Two blockchains are used in our design: one (BC-O) keeps tracks of the ownership history, the other (BC-P) keeps track of the play history. The play history chain is used to create undeniable uniqueness of each piece of content received by each user, and the ownership history chain enables the fee allocation as well as the transaction of articles. The structure of the blocks in the two chains are shown in Figure 2. The two blockchains are both public.

Each new block added must receive approval from the platform in the form of a digital signature. The transaction is realized via smart contracts. Whenever a new block with timestamp T is added to BC-P, the block on BC-O with timestamp closest to T will be the one used in determining the current owner(s) of the content. This set of owners is then the destination of the fee transfer. Each viewer on the platform has an asymmetry key pair, and the public key is published on the block in BC-P.

It should be note that, as opposed to Bitcoin or Ethereum, our system is not decentralized.

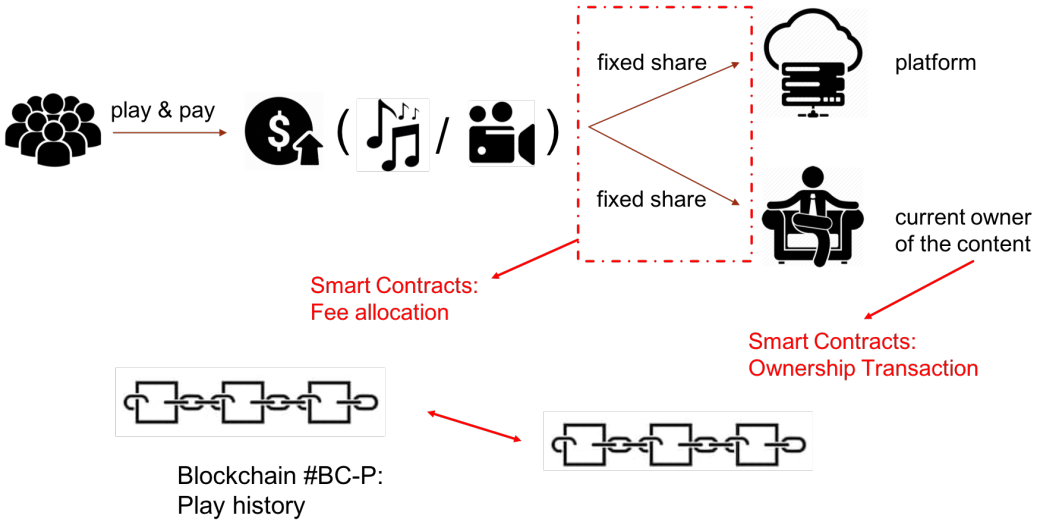

Figure 3.1: The basic idea of our platform design.

The platform must sign off on each new block on both BC-O and BC-P. However, the design prevents the existence of an adversarial platform that claims fake lawsuits against content leakages; the platform cannot arbitrarily condemn a user to have committed a content leakage (see sections 4.1 and 4.2).

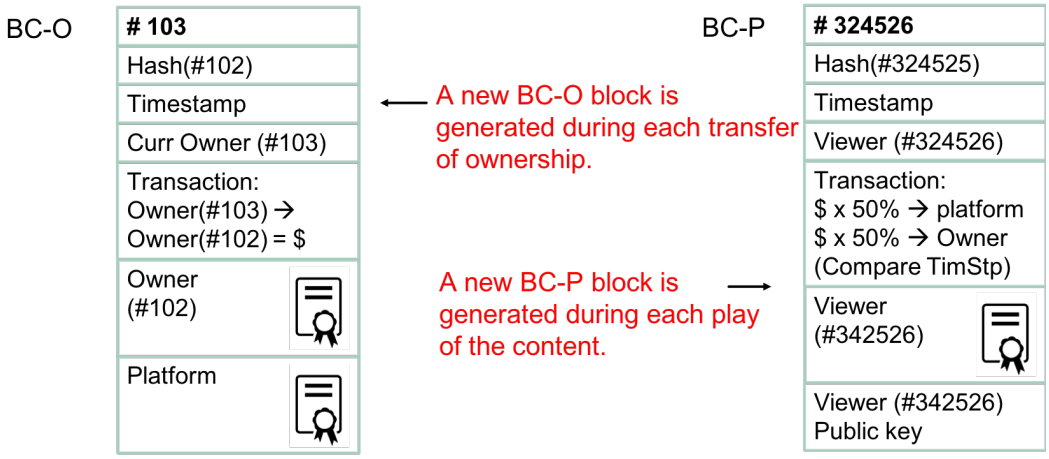

Figure 3.2: Example blocks of the two blockchains: ownership history chain (BC-O) and player history chain (BC-P).

Chapter 4

Possible Attacks and Defense

We now present several possible attacks on our blockchain-based system, each followed by ways to patch our system to prevent these attacks. Sections 4.1-4.4 each present possible attacks by each party involved in our system, and Section 4.5 presents possible unresolved issues that we intend to address in the future.

4.1 Adversarial Content Viewer: Content Leakage

In order for viewers to be able to play content provided by our proposed platform, they must be able to receive the content in a form that is close enough to the original to be playable. However, if the platform simply gives the viewer the original form of content when a new block is added, the viewer may upload the content as their own when it truly is owned by someone else.

This content leakage is prevented by adding watermarks to the contents. Each viewer will receive a content with an different but indistinguishable watermark. This process is guaranteed by steganography [3]. An illustration of our watermarking procedure is given in Figure 3.

Observe that the watermark for block $\#p$ depends on the value of p. That is, with a secret key, the platform should easily be able to retrieve both the original form of the content as well as the watermark contents and be able to determine which value of p the watermark corresponds to. If leaked content is detected in the ownership chain, the platform will be able to determine via this process which user leaked the content, and proceed to block the offending user's account.

4.2 Adversarial Platform: Undeniability of Leakage

To prevent a platform that claims leakage in an adversarial manner, viewers must have the right to contest a claim of leakage. The undeniability of leakage is guaranteed by the watermark design and explained as follows.

When a viewer wants to view content C, they propose a new block $\#p$, make a signature $sign(sk\#p, \#p)$ and send it to the platform. As explained in Figure 3, upon receiving this signature, the platform makes a watermark $h(\#p||sign(sk\#p, \#p))$, adds it to the original content, and sends the watermarked content to $\#p$ to watch.

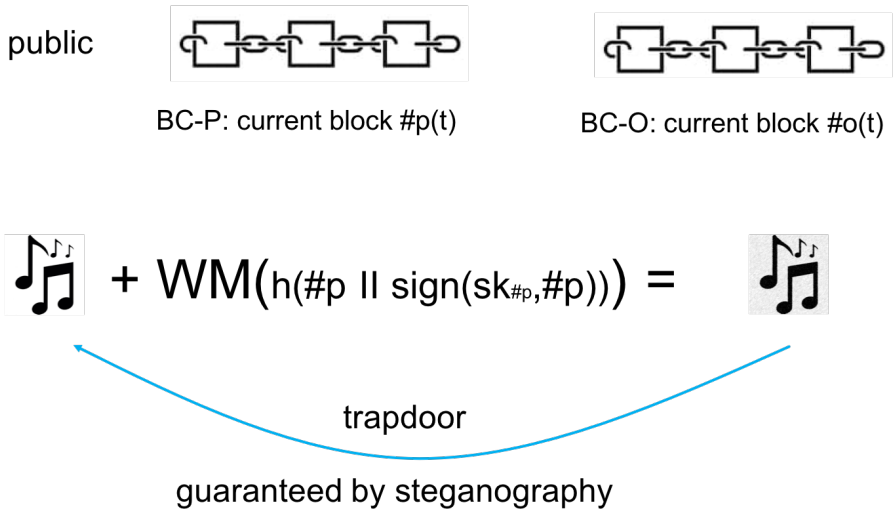

Figure 4.1: Adding watermarks to the digital contents.

Once leaked content is detected, the platform will reveal its watermark $h(\#p||sign(sk\#p, \#p))$. The platform will show to the public that $\#p||sign(sk\#p, \#p)$ is in the hash, and thus $\#p$ is the source of leakage.

People will verify that $sign(sk\#p, \#p)$ is from $\#p$ using his public key in the BC-P chain, and since the platform is unable to fake a signature from another $\#p'$, the platform will then not be able to claim a fake leakage.

4.3 Adversarial Content Provider: Repetitive Upload

The content provider might also be adversarial—one thing a content provider could do to harm the system is to upload multiple pieces of similar content solely for monetary gain. To prevent this type of attack, the platform will conduct similarity checks on the uploaded contents and make sure that similar pieces will be denied submission. This is similar to the situations described in sections 4.1 and 4.2 regarding content leakage detection. However, it is more difficult to determine similarity in this case, as there is no longer a guarantee that the changes made to the digital content will form any sort of identifiable watermark.

4.4 Content Owner

The content owner has no incentive to harm the system and he will actively stand out to protect the platform to make sure he can gather fees correctly. He may deny future transfer of ownership; but this all depends on the market. It is also possible to impose restrictions on ownership transfer that encourage transactions.

4.5 Flaws of Current Design

We anticipate there to be two main flaws with our current system.

1. A new BC-P block must be added for every "play" action a viewer initiates. Our system inconveniences the user in the sense that a user must create a digital signature each time they view a piece of digital content. In addition, since the platform must also sign off on all blocks, if these signatures by the platform are behaving in an unexpected manner, then digital content view requests may not be processed at all.
2. The similarity checks described in section 4.3 could be subject to machine-learning attacks. For example, even though certain small changes could be detected by a similarity-checking algorithm, others may not.

Chapter 5

Implementation and Future Work

5.1 Distributed App Development

To implement our DAPP (Distributed Application), we used Solidity [8], a contract-oriented language for Ethereum-based smart contracts, as our programming language. ERC20, a standard used for smart contracts on the Ethereum, is imported to support the basic features of tokens, which are the values to trade copyright ownership and rent payment.

To test our smart contract, we use Truffle [9] as our development environment and testing framework for Ethereum, which we can use to initialize test frameworks, and migration that is the deployment configuration. Instead of directly deploying our contract on Ethereum platform, we use Ganache [6], a personal blockchain, to deploy and test with simulated addresses.

5.2 CopyrightChain API

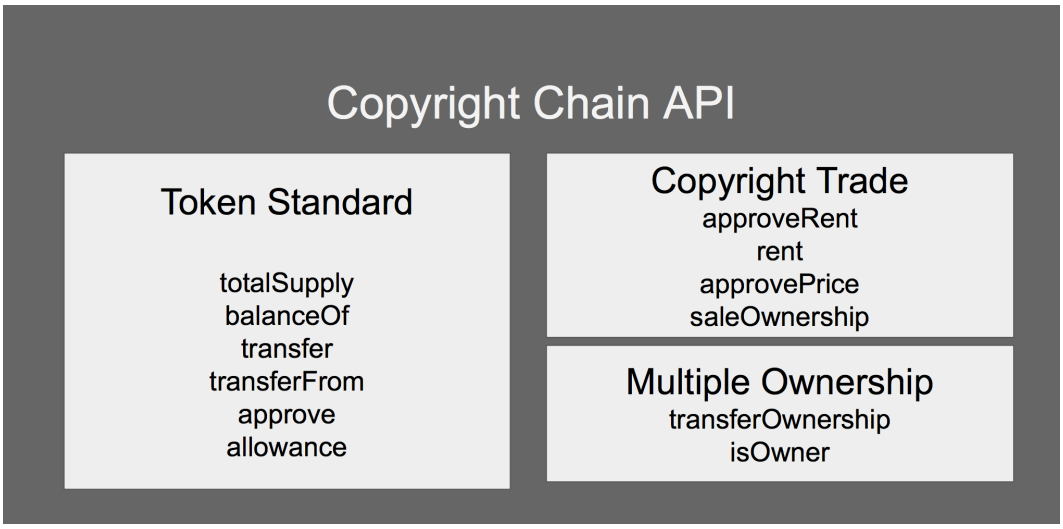

Figure 5.1: Alpha Version Implementation

Now, we have the the alpha version implementation that is submitted for review. The implementation is based on $ERC20$ library, which supports the initialization and balance transfer between accounts [11]. As $totalSupply$ and $balanceOf$ are used to query the total balance of the account and that of each individual, accounts can use $transfer$ tokens to others. By using $approve$, an account can authorize another account to withdraw at most a certain amount of tokens with $transferFrom$, $Allowance$ is called to query the allowable tokens.

Another important feature of our implementation is that the ownership the ownership of a copyright can be claimed by several different accounts. In the alpha version, every owner's share is uniform, and copyright is divided into a fix number of shares. When owners trade their shares and transfer their shares, they can only transfer all or none. This feature can be extended to support owning arbitrary portion of shares, and trade arbitrary shares they have. However, fixing the number of shares prevents the the explosion of owners, which results in low efficiency in voting.

With the token contract and multiple ownership feature, we are going to introduce the copyright trade module. A user can use $rent$ to rent the protected copyright of this chain with a price. Meanwhile, all the owners needs to agree on this price. Then, a watermarked file is released to this user, which is introduced in Chapter 3. The copyright share is also tradable. An owner can use $saleOwnership$ to trade its share to another user at a price approved by the owner. To approve a lower bound of rent, owners can call the function $approveRent$ to change price. To approve a lower bound price to trade an owner's ownership, it can can call $approvePrice$.

5.3 Experiments

```
  Contract: CopyrightChain
----------------------
Test 1
Balance of Owner:  10000
    ✓ should put 10000 Coins in the first account
----------------------
Test 2
Count =   3
    ✓ shoud have 3 owner
----------------------
Test 3
    ✓ Initial Ownership is correct (45ms)
----------------------
Test 4
    ✓ Ownership is transferred correctly (43ms)
----------------------
Test 5
    ✓ Fees are charged correctly


  5 passing (189ms)
```

Figure 5.2: The basic idea of our platform design.

To demonstrate the basic functionality of our implementation, we design the following experiments illustrated in Figure 5.2:

- 1. When the contract is created, each account obtains correct amount of tokens.
- 2 and 3. Meanwhile, we specify three owners, which are $address(0)$, $address(1)$ and $address(3)$.
- 4. $address(1)$ transfer its ownership to $address(2)$.
- 5. $address(1)$ rent the copyright from $address(0)$, $address(2)$ and $address(3)$, and fees are charged correctly.

5.4 Future Implementation

Our current implementation is limited to the blockchain and contract structure—we have not yet built the portion of the platform that interacts with users. Namely, the following two features have not yet been implemented:

- We still have not yet connected our contract system with actual database of digital content. Upon purchase of digital content, the platform should automatically release the corresponding content in a database to the user, as described in Chapter 3. The content should be authenticated with a code that contains the most recent timestamp on the ownership chain, as well as the user address. In addition, as described in section 4.1, there must be a watermark applied to the content that must be easily invertible by the platform but not by the user.
- We also have not yet extended our ownership transfer routine to support uniform sharing. Namely, if the original contract has four owners for digital content C, each should have an equal share of C, and ownership transfers must be all-or-nothing.

References

Literature

[1] *Blockchain: Use Case - Copyright*. www.limegreenipnews.com/2017/11/blockchain-use-case-copyright/. Accessed: 2018-03-22 (cit. on p. 3).

[2] Vitalik Buterin et al. "Ethereum white paper". In: *GitHub repository* (2013) (cit. on p. 2).

[3] Ingemar Cox et al. *Digital watermarking and steganography*. Morgan Kaufmann, 2007 (cit. on p. 8).

[4] *Create your own Crypto-Currency with ethereum*. www.ethereum.org/token. Accessed: 2018-03-22 (cit. on p. 2).

[5] *Ethereum Tokens Tracker*. www.etherscan.io/tokens. Accessed: 2018-03-22 (cit. on p. 2).

[6] *Ganache*. http://truffleframework.com/docs/ganache/using. Accessed: 2018-03-22 (cit. on p. 11).

[7] Satoshi Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". In: (2008) (cit. on p. 2).

[8] *Solidity*. https://solidity.readthedocs.io/. Accessed: 2018-03-22 (cit. on p. 11).

[9] *Truffle*. http://truffleframework.com/. Accessed: 2018-03-22 (cit. on p. 11).

[10] *ujo Music*. www.ujomusic.com. Accessed: 2018-03-22 (cit. on p. 5).

[11] F Vogelsteller. *ERC 20 token standard*. 2015 (cit. on pp. 2, 12).